Theoretical Computer Science 410 (2009) 3597–3605

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Online parallel machines scheduling with two hierarchies

An Zhang [a], Yiwei Jiang [b], Zhiyi Tan [a,*]

[a] Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, PR China

[b] Faculty of Science, Zhejiang Sci-Tech University, Hangzhou 310018, PR China

ARTICLE INFO

Article history:
Received 28 August 2008
Received in revised form 28 March 2009
Accepted 8 April 2009
Communicated by D.-Z. Du

Keywords:
Scheduling
Online
Hierarchical machines
Competitive ratio

ABSTRACT

This paper investigates an online hierarchical scheduling problem on m parallel identical machines. Each job, as well as each machine, has a hierarchy associated with it. A job can be scheduled on a machine only when its hierarchy is no higher than that of the machine. The objective is to minimize the makespan. In addition, we assume that there are only two hierarchies, and k machines have a higher hierarchy which can schedule all jobs. We present an online algorithm with a competitive ratio of $1 + \frac{m^2-m}{m^2-km+k^2} < \frac{7}{3}$ for any k and m. The performance for some pairs of k and m is further improved by another algorithm. Lower bounds for various pairs of k and m are also presented.

1. Introduction

Problem statement. In this paper we investigate an online hierarchical scheduling problem on parallel identical machines. Jobs arrive one by one over list and each may be scheduled only on a special subset of the machines. Namely, a job j has a size p_j and a hierarchy g_j associated with it, and a machine M_i is also associated with a hierarchy $g(M_i)$. The job j can be scheduled on the machine M_i only if $g_j \geq g(M_i)$. The goal is to assign all jobs to the permitted machines so as to minimize the makespan, i.e., the maximum completion time of all machines.

The performance of an algorithm A for an online problem is often evaluated by its competitive ratio, which is defined as the smallest number ρ such that for any job sequence $\mathcal{J}$, $C^A(\mathcal{J}) \leq \rho C^*(\mathcal{J})$, where $C^A(\mathcal{J})$ (or in short C^A) denotes the makespan produced by A and $C^*(\mathcal{J})$ (or in short C^*) denotes the optimal makespan in an offline version. Due to the very nature of online scheduling, it is often possible to prove that there is a lower bound to the competitive ratio achievable by any deterministic online algorithm.

In this paper, we consider the online version of the problem on m identical machines with two hierarchies, that is, $g(M_i)$ is equal to 1 or 2 for all $1 \leq i \leq m$, and g_j is also equal to 1 or 2 for all j. Such an assumption is reasonable and valuable, because in real applications, the hierarchy setting will not be very complicated. In many cases, exactly two hierarchies will be classified. For example, in service industries, customers are usually divided into VIP and ordinary. The memory of computer systems is usually divided into fast access memory and slow memory, etc. Without loss of generality, we assume that $g(M_i) = 1$ for $i = 1, \ldots, k$ and $g(M_i) = 2$ for $i = k+1, \ldots, m$, where $0 \leq k \leq m$. If $k = 0$ or $k = m$, then the problem can be reduced to the classical online scheduling problem on parallel identical machines with the objective of minimizing the makespan [1,6–8,15]. Therefore, we suppose $1 \leq k \leq m-1$ in this paper.

Related works. One related but more general model is *restricted assignment* [2] proposed by Azar et al., where each job has a subset of machines on which it may be scheduled. They presented an online algorithm with a competitive ratio $\lceil \log_2 m \rceil + 1$ for any m.

* Corresponding author. Tel.: +86 571 87951429; fax: +86 571 87953715.
E-mail addresses: anzhanghz@yahoo.com.cn (A. Zhang), mathjyw@yahoo.com.cn (Y. Jiang), tanzy@zju.edu.cn (Z. Tan).

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2009.04.007

Table 1
Upper and lower bounds for small m and k.

m	Online scheduling		Hierarchical online scheduling			
	$k=0$		$k=1$		$k=2$	
	LB	UB	LB	UB	LB	UB
2	1.5 [7]	1.5 [10]	1.667 [13,14]	1.667 [13,14]		
3	1.667 [7]	1.667 [10]	1.824	1.857 (*TLS*)	1.801	1.857 (*TLS*)
4	1.732 [15]	1.733 [6]	1.848	1.923 (*TLS*)	1.907	2 (*SLS*)
5	1.746 [6]	1.771 [6]	1.848	1.952 (*TLS*)	1.907	2 (*SLS*)
6	1.773 [6]	1.8 [6]	1.829	1.968 (*TLS*)	1.907	2 (*SLS*)

For the hierarchical model on m machines with general hierarchies, Bar-Noy et al. [3] designed a non-preemptive $e+1 \approx 3.718$-competitive algorithm (also in [5]), which is also showed to be e-competitive when all jobs have unit size. In addition, Hwang et al. [11] considered the offline version of the problem and presented an algorithm with a worst case ratio no more than $5/4$ for $m=2$ and $2-1/(m-1)$ for $m \geq 3$. Glass and Kellerer [9] gave an improved algorithm with a worst-case ratio at most $3/2$ for m machines.

Recently, Jiang et al. [13] and Park et al. [14] independently presented an optimal online algorithm with a competitive ratio of $5/3$ on two identical machines. Afterwards, [12] extended the result to the general case that there are exactly two hierarchies on m machines, i.e., the problem considered in this paper. It is shown that the lower bound when $m=16$ and $k=4$ is at least 2 and the greedy algorithm has a competitive ratio of $4-1/m$. Besides, an improved online algorithm with a competitive ratio of $\frac{12+4\sqrt{2}}{7} \approx 2.522$ was proposed.

In [14], a semi-online model with known total size of all jobs was considered and the authors presented an optimal algorithm with a competitive ratio of $3/2$ on two identical machines. [13] studied the preemptive model in which idle time is not allowed and the authors presented an optimal algorithm with a competitive ratio of $3/2$ on two identical machines. In [4], Chassid and Epstein extended the hierarchial scheduling model to two uniform machines, online and semi-online problems were studied and optimal algorithms were proposed.

Our results. We consider the online scheduling problem with two hierarchies on m identical machines. We first design an algorithm *TLS* with a competitive ratio of $1+\frac{m^2-m}{m^2-km+k^2} < \frac{7}{3}$, which improves the result in [12]. A relatively simple algorithm *SLS* is also given, whose performance is better for some pairs of k and m. We further give a comprehensive study on lower bounds of the problem for different pairs of k and m. Results of lower bounds and the competitive ratio of online algorithms for small m and k, compared with those of classical online scheduling, can be found in Table 1. We also prove the lower bound is 2 when $k \geq 3$ and $m \geq \frac{3}{2}(k+1)$. These lower bounds are greater than the lower bounds on the classical online scheduling problem.

The rest of the paper is organized as follows. Section 2 gives some basic notations and useful lemmas. Sections 3 and 4 consider the online algorithms and lower bounds, respectively. Finally, some conclusions will be made in Section 5.

2. Preliminary

The following notations and definitions are used in the remainder of the paper.

n	The number of jobs.
$\mathcal{M}_i$	The set of machines with hierarchy i.
V_i	The set of jobs with hierarchy i.
T_j	The total size of the first j jobs.
T_{ji}	The total size of the jobs with hierarchy i in the first j jobs.
$p_j^{\max}$	The largest job size in the first j jobs.
L_j^i	The completion time of machine M_i after the j-th job is processed in a schedule generated by an online algorithm A.
L_j^*	The optimal makespan of the sequence containing the first j jobs.

Let

$$LB_j = \max\left\{p_j^{\max}, \frac{T_j}{m}, \frac{T_{j1}}{k}\right\}, \quad j=1,\ldots,n. \tag{1}$$

Clearly LB_j is a nondecreasing function of j. In the following, we first show a lower bound on the optimal makespan. Next, a simple but useful property is given.

Lemma 2.1. *$L_j^* \geq LB_j$, for any $j \geq 1$.*

Proof. Obviously, $L_j^* \geq \max\{p_j^{\max}, \frac{T_j}{m}\}$ for any $j \geq 1$. Note that all jobs in V_1 can be only processed on $\mathcal{M}_1$, which implies that $L_j^* \geq \frac{T_{j1}}{k}$. Hence the result follows. □

Lemma 2.2. *Let* $\mathbf{1} = (1, 1, \ldots, 1)^{\mathrm{T}}$, $\mathbf{x} = (x_1, x_2, \ldots, x_q)^{\mathrm{T}}$ *be* $q \times 1$ *matrices and* $\mathbf{c} = (c_1, c_2, \ldots, c_q)$ *be a* $1 \times q$ *matrix.* $\mathbf{A} = (a_{ij})_{q\times q}$ *is an invertible matrix, and the i-th row vector of* $\mathbf{A}$ *is denoted as* $\alpha_{\mathbf{i}}$. *If* $\mathbf{cA^{-1}} \geq \mathbf{0}$, *then* $\mathbf{cx} \leq (\mathbf{cA^{-1}1}) \max\{\alpha_{\mathbf{1}}\mathbf{x}, \alpha_{\mathbf{2}}\mathbf{x}, \ldots, \alpha_{\mathbf{q}}\mathbf{x}\}$ *for any* $\mathbf{x} \geq \mathbf{0}$.

Proof. As $\mathbf{A}$ is invertible, it is clear that $\mathbf{cx} = (\mathbf{cA^{-1}})(\mathbf{Ax})$. Moreover, by $\mathbf{cA^{-1}} \geq \mathbf{0}$, we can conclude that $\mathbf{cx} = (\mathbf{cA^{-1}})(\alpha_{\mathbf{1}}\mathbf{x}, \alpha_{\mathbf{2}}\mathbf{x}, \ldots, \alpha_{\mathbf{q}}\mathbf{x})^{\mathrm{T}} \leq (\mathbf{cA^{-1}1}) \max\{\alpha_{\mathbf{1}}\mathbf{x}, \alpha_{\mathbf{2}}\mathbf{x}, \ldots, \alpha_{\mathbf{q}}\mathbf{x}\}$. □

It is well known that *LS* (*List Scheduling*) was originally designed by Graham [10] for online scheduling on identical machines. It schedules jobs one by one on the least loaded machine. Though *LS* cannot be directly used for hierarchical scheduling, we adopt the *LS* rule as subprocedures of our algorithms. Let $\mathcal{M}$ be a subset of the machine set, *schedule job j on* $\mathcal{M}$ *by LS rule* implies that job j is scheduled on the machine which has the smallest load of $\mathcal{M}$ before assigning the job j.

3. Online algorithms

In this section, we consider online algorithms for any pairs of k and m. For simplicity, we identify the jobs with their sizes. We first present an online algorithm *Threshold LS* (*TLS* for short), which schedules any job of V_2 on $\mathcal{M}_2$ unless all these machines would exceed a certain load. Write $\alpha = \frac{m^2-m}{m^2-km+k^2}$. Note that

$$\frac{k-1}{k} < \frac{m-1}{m} < \alpha < \frac{m}{m-k} \tag{2}$$

holds for any k and m.

Algorithm *TLS*

1. Initially set $L_0^i = 0$ for all $i = 1, 2, \ldots, m$ and let $j = 1$.
2. If $p_j \in V_1$, schedule p_j on $\mathcal{M}_1$ by the *LS* rule.
3. If $p_j \in V_2$, let $t \in \arg\min\{L_{j-1}^i | i = k+1, \ldots, m\}$. If $L_{j-1}^t + p_j \leq (1+\alpha)LB_j$, schedule p_j on M_t. Otherwise, schedule p_j on $\mathcal{M}_1$ by the *LS* rule.
4. If no new job arrives, stop. Otherwise, $j = j + 1$, return step 2.

W.l.o.g., suppose that the last job p_n determines the makespan. Let s_n be the time at which p_n starts, i.e., $C^{TLS} = p_n + s_n$.

Theorem 3.1. *For any given k and m, the competitive ratio of algorithm TLS is* $1 + \alpha = 1 + \frac{m^2-m}{m^2-km+k^2}$.

Proof. We distinguish two cases according to the hierarchy of p_n.

Case 1 $p_n \in V_2$.

By the definition of *TLS*, if p_n is scheduled on one machine in $\mathcal{M}_2$, say M_t, $k+1 \leq t \leq m$, then $C^{TLS} = L_{n-1}^t + p_n \leq (1+\alpha)LB_n$. By Lemma 2.1, we have $C^{TLS} \leq (1+\alpha)C^*$. We are left to consider the situation that p_n is scheduled on $\mathcal{M}_1$. By the *LS* rule and algorithm *TLS*, we have

$$s_n \leq \frac{1}{k}\sum_{i=1}^{k} L_{n-1}^i \tag{3}$$

and $L_{n-1}^i + p_n > (1+\alpha)LB_n$ for any $i = k+1, \ldots, m$. Together with Lemma 2.1 and (1), we obtain

$$\begin{aligned} C^{TLS} &= p_n + s_n \leq p_n + \frac{1}{k}\sum_{i=1}^{k} L_{n-1}^i = p_n + \frac{1}{k}\left(T_n - p_n - \sum_{i=k+1}^{m} L_{n-1}^i\right) \\ &= \frac{m-1}{k}p_n + \frac{1}{k}T_n - \frac{1}{k}\sum_{i=k+1}^{m}(p_n + L_{n-1}^i) < \frac{m-1}{k}LB_n + \frac{m}{k}LB_n - \frac{(m-k)(1+\alpha)}{k}LB_n \\ &= \frac{m+k-1-(m-k)\alpha}{k}LB_n \leq \frac{m+k-1-(m-k)\alpha}{k}C^* < (1+\alpha)C^*, \end{aligned}$$

where the last inequality is due to (2).

Case 2 $p_n \in V_1$.

If all the jobs processed on $\mathcal{M}_1$ are from V_1, then $s_n \leq \frac{T_{n1}-p_n}{k}$, since p_n is assigned to $\mathcal{M}_1$ by the *LS* rule. By Lemma 2.1 and (1),

$$C^{TLS} = s_n + p_n \leq \frac{1}{k}T_{n1} + \left(1 - \frac{1}{k}\right)p_n \leq \left(2 - \frac{1}{k}\right)LB_n \leq \left(2 - \frac{1}{k}\right)C^* \leq (1+\alpha)C^*,$$

where the last inequality is due to (2). Otherwise, suppose p_j is the last job in V_2 which is scheduled on $\mathcal{M}_1$. In other words, after p_j is scheduled, all jobs scheduled on $\mathcal{M}_1$ are from V_1. Hence,

$$T_{n1} \geq \left(p_n + \sum_{i=1}^{k} L_{n-1}^i\right) - \left(p_j + \sum_{i=1}^{k} L_{j-1}^i\right). \tag{4}$$

Since *TLS* does not schedule p_j on machines in $\mathcal{M}_2$, $L_{j-1}^i + p_j > (1+\alpha)LB_j$ for any $i = k+1, \ldots, m$. Together with (1), we have

$$\sum_{i=k+1}^{m} L_{j-1}^i > \sum_{i=k+1}^{m} \left((1+\alpha)LB_j - p_j\right) \geq \sum_{i=k+1}^{m} \alpha LB_j \geq \sum_{i=k+1}^{m} \alpha \frac{T_j}{m} = \frac{(m-k)\alpha}{m} T_j$$

and

$$p_j + \sum_{i=1}^{k} L_{j-1}^i = T_j - \sum_{i=k+1}^{m} L_{j-1}^i < \frac{m}{(m-k)\alpha} \sum_{i=k+1}^{m} L_{j-1}^i - \sum_{i=k+1}^{m} L_{j-1}^i \leq \frac{m-(m-k)\alpha}{(m-k)\alpha} \sum_{i=k+1}^{m} L_n^i. \tag{5}$$

Since the algorithm schedules p_n on $\mathcal{M}_1$ by the *LS* rule, inequality (3) still holds. Substitute (5) and (3) into (4), we obtain

$$T_{n1} \geq p_n + ks_n - \frac{m-(m-k)\alpha}{(m-k)\alpha} \sum_{i=k+1}^{m} L_n^i. \tag{6}$$

Moreover, by (3) and the assumption of $p_n \in V_1$,

$$T_n \geq \sum_{i=1}^{k} L_n^i + \sum_{i=k+1}^{m} L_n^i = p_n + \sum_{i=1}^{k} L_{n-1}^i + \sum_{i=k+1}^{m} L_n^i \geq p_n + ks_n + \sum_{i=k+1}^{m} L_n^i. \tag{7}$$

By Lemma 2.1 and (1), (6) and (7), we have

$$C^* \geq LB_n \geq \max\left\{p_n, \frac{p_n + ks_n + \sum_{i=k+1}^{m} L_n^i}{m}, \frac{p_n + ks_n - \frac{m-(m-k)\alpha}{(m-k)\alpha} \sum_{i=k+1}^{m} L_n^i}{k}\right\}.$$

Let $\mathbf{c} = (1, 1, 0)$, $\mathbf{x} = (p_n, s_n, \sum_{i=k+1}^{m} L_n^i)^{\mathrm{T}}$ and

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{m} & \frac{k}{m} & \frac{1}{m} \\ \frac{1}{k} & 1 & \frac{(m-k)\alpha - m}{(m-k)k\alpha} \end{pmatrix}.$$

It can be verified directly that $\mathbf{cx} = p_n + s_n$, $C^* \geq \max\{\alpha_1\mathbf{x}, \alpha_2\mathbf{x}, \alpha_3\mathbf{x}\}$ and

$$\mathbf{cA^{-1}} = (1, 1, 0) \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{k} & \frac{m-(m-k)\alpha}{k} & \frac{(m-k)\alpha}{m} \\ 0 & (m-k)\alpha & -\frac{k(m-k)\alpha}{m} \end{pmatrix} = \left(1 - \frac{1}{k}, \frac{m-(m-k)\alpha}{k}, \frac{(m-k)\alpha}{m}\right).$$

Consequently $\mathbf{cA^{-1}} \geq \mathbf{0}$ by (2) and $\mathbf{cA^{-1}1} = 1 + \alpha$ by the definition of α. By Lemma 2.2, we have

$$C^{TLS} = p_n + s_n = \mathbf{cx} \leq (\mathbf{cA^{-1}1}) \max\{\alpha_1\mathbf{x}, \alpha_2\mathbf{x}, \alpha_3\mathbf{x}\} \leq (1+\alpha)C^*.$$

The bound is tight for any pairs of k and m. If $k = 1$ or $k = m-1$, consider the sequence with $m-k$ jobs of size $m^2 - m$, followed by k jobs of size $m^2 - m + 1$. All jobs have hierarchy 2. Clearly, *TLS* must schedule the first $m-k$ jobs on $m-k$ machines in $\mathcal{M}_2$ on average. Since

$$T_j \leq T_n = (m-k)(m^2-m) + k(m^2-m+1) < m(m^2-m+1) = mp_j^{\max}$$

and

$$(m^2-m) + (m^2-m+1) = \left(1 + \frac{m^2-m}{m^2-m+1}\right)(m^2-m+1) = (1+\alpha)LB_j$$

for any $m-k+1 \leq j \leq m$, one of the last k jobs will be scheduled on some machine in $\mathcal{M}_2$. Hence, $C^{TLS} = 2m^2 - 2m + 1$. In the optimal schedule, each job is scheduled on a different machine, then $C^* = m^2 - m + 1$ and $\frac{C^{TLS}}{C^*} = 1 + \alpha$.

If $2 \le k \le m-2$, then $m \ge 4$ and $km - m - k^2 > 0$. Consider the sequence with $m-k$ jobs of size $km - m - k^2$, followed by $m-k+1$ jobs of size $m^2 - km + k^2$. All jobs have hierarchy 2. *TLS* still schedule the first $m-k$ jobs on $m-k$ machines in $\mathcal{M}_2$ on average. Since

$$T_j \le T_n = (m-k)(km - m - k^2) + (m-k+1)(m^2 - km + k^2) < m(m^2 - km + k^2) = mp_j^{\max}$$

and

$$(km - m - k^2) + 2(m^2 - km + k^2) = \left(1 + \frac{m^2 - m}{m^2 - km + k^2}\right)(m^2 - km + k^2) = (1+\alpha)LB_j$$

for any $m-k+1 \le j \le 2m-2k+1$, *TLS* also schedules the first $m-k$ jobs of size $m^2 - km + k^2$ on $m-k$ machines in $\mathcal{M}_2$ averagely. The last job will be scheduled on one machine in $\mathcal{M}_2$. Hence, $C^{TLS} = 2m^2 - (k+1)m + k^2$. In the optimal schedule, jobs of size $m^2 - km + k^2$ are scheduled on $m-k+1$ machines separately. Jobs of size $km - m - k^2$ are scheduled on the remaining $k-1$ machines on average. Thus

$$C^* = \max\left\{\left\lceil \frac{m-k}{k-1} \right\rceil (km - m - k^2), m^2 - km + k^2\right\} = m^2 - km + k^2$$

and $\frac{C^{TLS}}{C^*} = 1 + \alpha$. □

Note that

$$1 + \frac{m^2 - m}{m^2 - km + k^2} = 1 + \frac{1 - \frac{1}{m}}{(\frac{k}{m} - \frac{1}{2})^2 + \frac{3}{4}} < \frac{7}{3},$$

so the algorithm *TLS* has a smaller competitive ratio than the previous algorithms. On the other hand, *TLS* is unsatisfactory for some pairs of k and m. Hence, we design another algorithm *Split LS* (*SLS* for short) for $k \ge 2$ in the following.

Let $l_0 = \frac{m(k-1)}{m+k-2}$ and

$$l = \begin{cases} \lfloor l_0 \rfloor, & \text{if } \dfrac{k-1}{\lfloor l_0 \rfloor} \le \dfrac{m-1}{m - \lceil l_0 \rceil}, \\ \lceil l_0 \rceil, & \text{otherwise.} \end{cases}$$

Therefore, the definition of l ensures that $l \le \lceil l_0 \rceil \le l_0 + 1 = \frac{m(k-1)}{m+k-2} + 1 \le k$ and $l \ge \lfloor l_0 \rfloor = \lfloor \frac{m(k-1)}{m+k-2} \rfloor \ge 1$. Moreover,

$$\max\left\{\frac{k-1}{l}, \frac{m-1}{m-l}\right\} = \min\left\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m - \lceil l_0 \rceil}\right\}. \tag{8}$$

In fact, if $\frac{k-1}{\lfloor l_0 \rfloor} \le \frac{m-1}{m - \lceil l_0 \rceil}$, then $l = \lfloor l_0 \rfloor \le l_0 = \frac{m(k-1)}{m+k-2}$. Hence,

$$\max\left\{\frac{k-1}{l}, \frac{m-1}{m-l}\right\} = \frac{k-1}{l} = \frac{k-1}{\lfloor l_0 \rfloor} = \min\left\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m - \lceil l_0 \rceil}\right\}.$$

The case of $\frac{k-1}{\lfloor l_0 \rfloor} > \frac{m-1}{m - \lceil l_0 \rceil}$ can be proved similarly.

Algorithm *SLS*

1. Let $j = 1$.
2. If $p_j \in V_1$, schedule p_j on $\{M_1, M_2, \ldots, M_l\}$ by the *LS* rule.
3. If $p_j \in V_2$, schedule p_j on $\{M_{l+1}, \ldots, M_m\}$ by the *LS* rule.
4. If no new job arrives, stop. Otherwise, $j = j + 1$, return step 2.

In fact, *SLS* splits machine set into two subsets. One of them is a subset of $\mathcal{M}_1$. Jobs of two hierarchies are scheduled on two subsets separately simply by the *LS* rule.

Theorem 3.2. *Algorithm SLS has a competitive ratio of* $1 + \min\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m - \lceil l_0 \rceil}\}$ *for any* m *and* $2 \le k \le m-1$.

Proof. W.l.o.g., suppose p_n determines the makespan. Then we have $C^{SLS} = p_n + s_n$. If $p_n \in V_1$, then $T_{n1} \ge p_n + ls_n$. It follows that $C^* \ge LB_n \ge \max\{p_n, \frac{p_n + ls_n}{k}\}$ by Lemma 2.1 and (1). Let $\mathbf{c} = (1, 1)$, $\mathbf{x} = (p_n, s_n)^{\mathrm{T}}$ and $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ \frac{1}{k} & \frac{l}{k} \end{pmatrix}$, then $\mathbf{cA^{-1}} = (1, 1)\begin{pmatrix} 1 & 0 \\ -\frac{1}{l} & \frac{k}{l} \end{pmatrix} = (1 - \frac{1}{l}, \frac{k}{l})$. We can obtain

$$C^{SLS} = \mathbf{cx} \le (\mathbf{cA^{-1}1}) \max\{\alpha_1 \mathbf{x}, \alpha_2 \mathbf{x}\} \le \left(1 + \frac{k-1}{l}\right) C^*$$

Table 2(a)
Illustration for the layers used in Theorem 4.1.

Layer	Job hierarchy	Number of jobs		Job size
		$m \le 2k$	$m > 2k$	
I	2	m	m	1
II	1	m	$2k$	1
III	1	$2k-m+1$	1	3

by Lemma 2.2. If $p_n \in V_2$, we have $T_n \ge T_{n2} \ge p_n + (m-l)s_n$. Thus $C^* \ge LB_n \ge \max\{p_n, \frac{p_n+(m-l)s_n}{m}\}$ by Lemma 2.1 and (1). Similarly, let $\mathbf{c} = (1, 1)$, $\mathbf{x} = (p_n, s_n)^{\mathrm{T}}$ and $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ \frac{1}{m} & \frac{m-l}{m} \end{pmatrix}$, then $\mathbf{cA^{-1}} = (1, 1)\begin{pmatrix} 1 & 0 \\ -\frac{1}{m-l} & \frac{m}{m-l} \end{pmatrix} = (1 - \frac{1}{m-l}, \frac{m}{m-l})$. We can obtain

$$C^{SLS} = \mathbf{cx} \le (\mathbf{cA^{-1}1}) \max\{\alpha_1\mathbf{x}, \alpha_2\mathbf{x}\} \le \left(1 + \frac{m-1}{m-l}\right) C^*$$

by Lemma 2.2. Hence, by (8),

$$\frac{C^{SLS}}{C^*} \le 1 + \max\left\{\frac{k-1}{l}, \frac{m-1}{m-l}\right\} = 1 + \min\left\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m-\lceil l_0 \rceil}\right\}.$$

The following job sequences show that the bound is tight for any pairs of m and k. If $\frac{k-1}{l} > \frac{m-1}{m-l}$, the sequence contains $l(k-1)$ jobs of size $\frac{1}{l}$, followed by one job with size 1. All jobs have hierarchy 1. Clearly, $C^{SLS} = \frac{k-1}{l} + 1$, $C^* = 1$ and $\frac{C^{SLS}}{C^*} = 1 + \frac{k-1}{l}$. If $\frac{k-1}{l} \le \frac{m-1}{m-l}$, the sequence contains $(m-l)(m-1)$ jobs of size $\frac{1}{m-l}$, followed by one job with size 1. All jobs have hierarchy 2. Obviously, $C^{SLS} = \frac{m-1}{m-l} + 1$, $C^* = 1$ and $\frac{C^{SLS}}{C^*} = 1 + \frac{m-1}{m-l}$. □

Although *SLS* seems relatively simple, it definitely beats *TLS* for some pairs of k and m. For example, when $k = 2$, *SLS* is 2-competitive for any $m \ge 3$, while the corresponding competitive ratio of *TLS* is greater than 2 if $m \ge 5$. For $3 \le k < \sqrt{m}$, we have $k - 2 < l_0 < k - 1$. Thus

$$1 + \min\left\{\frac{k-1}{\lfloor l_0 \rfloor}, \frac{m-1}{m-\lceil l_0 \rceil}\right\} = 1 + \frac{m-1}{m-\lceil l_0 \rceil} < 1 + \frac{m^2-m}{m^2-km+k^2},$$

SLS also performs better than *TLS*.

4. Lower bounds

The sequences of jobs which are used for establishing lower bounds are divided into successive layers, denoted by I, II, ⋯. Jobs inside a layer have the same size and hierarchy. The layers are constructed such that in order to perform well, any online algorithm has to schedule the first several layers in a special manner, and as a result, it cannot handle the jobs in the last layer (usually contains only one job with largest size and hierarchy 1) very well. On the other hand, in order to derive the upper bounds of the optimal makespan, feasible schedules, also for partial sequences which only contain first several layers, should be given. We demonstrate them by using a simple expression. For example, 3×{II, 2IV} implies that in the feasible schedule, there are 3 machines, each of which schedules one job of layer II and two jobs of layer IV. The completion time of each machine can be calculated accordingly.

Theorem 4.1. *For $k \ge 3$ and $m \ge \frac{3}{2}(k+1)$, any online algorithm has a competitive ratio of at least* 2.

Proof. If $\frac{3}{2}(k+1) \le m \le 2k$, we use the job sequence shown in Table 2(a). Clearly, in order to be better than 2-competitive, any algorithm has to schedule the jobs in layer I on m machines evenly. Then layer II comes in. Jobs in layer II have to be scheduled on k machines of $\mathcal{M}_1$. If there is some machine whose load achieves 4, then from Table 2(b), we get $C^* \le 2$ and $\frac{C^A}{C^*} \ge 2$. Hence, we assume that all machines in $\mathcal{M}_1$ have a load at most 3. Furthermore, since there are m jobs of size 1 in layer II to be scheduled on k machines in $\mathcal{M}_1$, at least $m-k$ machines must achieve the load 3. In other words, at most $2k-m$ machines of $\mathcal{M}_1$ have loads smaller than 3. Now layer III arrives. If no job in this layer is scheduled on those machines with load 3, then all $2k-m+1$ jobs in this layer have to be scheduled on at most $2k-m$ machines. At least two of them must be scheduled on the same machine, hence $C^A \ge 6$. If some jobs in this layer are scheduled on those machines with load 3, $C^A \ge 6$ trivially holds. On the other hand, we know $C^* \le 3$ from Table 2(b), thus the desired lower bound also holds. The case of $m > 2k$ can be proved similarly. □

Table 2(b)
Feasible solutions for partial layers.

Partial layers	A feasible schedule for $\frac{3}{2}(k+1) \le m \le 2k$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I	$k\times\{\mathrm{I}\}$	$(m-k)\times\{\mathrm{I}\}$	1
I, II	$(m-k)\times\{2\mathrm{II}\}$, $(2k-m)\times\{\mathrm{I}, \mathrm{II}\}$	$(m-k)\times\{2\mathrm{I}\}$	2
I, II, III	$(2m-3k-3)\times\{2\mathrm{II}\}$, $(2k+2-m)\times\{3\mathrm{II}\}$, $(2k-m+1)\times\{\mathrm{III}\}$	$(2k-m)\times\{3\mathrm{I}\}$, $(2m-3k)\times\{2\mathrm{I}\}$	3
Partial layers	A feasible schedule for $m > 2k$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I	$k\times\{\mathrm{I}\}$	$(m-k)\times\{\mathrm{I}\}$	1
I, II	$k\times\{2\mathrm{II}\}$	$(m-2k)\times\{\mathrm{I}\}$, $k\times\{2\mathrm{I}\}$	2
I, II, III	$\{\mathrm{III}\}$, $2\times\{3\mathrm{II}\}$, $(k-3)\times\{2\mathrm{II}\}$	$(m-2k)\times\{\mathrm{I}\}$, $k\times\{2\mathrm{I}\}$	3

Table 3(a)
Illustration for layers used in Theorem 4.2.

Layer	Job hierarchy	Number of jobs	Job size	
			$m=3$	$m\ge 4$
I	2	m	1	1
II	2	m	x_1	x_2
III	1	2	$\frac{5x_1^2+x_1-1}{2+2x_1}$	$\frac{3x_2^2-1}{2+2x_2}$
IV	1	1	$\frac{5x_1^2+x_1-1}{1+x_1}$	$\frac{3x_2^2-1}{1+x_2}$

Theorem 4.2. *For $k = 2$, any algorithm has a competitive ratio of at least*

$$\begin{cases} \dfrac{1+2x_1}{1+x_1} \approx 1.801, & m = 3; \\ \dfrac{1+2x_2}{1+x_2} \approx 1.907, & m \ge 4. \end{cases}$$

where $x_1 \approx 4.018$ and $x_2 \approx 9.730$ are the roots of the equations $3x^3 - 10x^2 - 8x - 1 = 0$ and $x^3 - 9x^2 - 7x - 1 = 0$ respectively.

Proof. We use sequences containing four layers in this proof. The number of jobs, the size and hierarchy of jobs in each layer can be found in Table 3(a). We only prove the case of $m \ge 4$ thoroughly, the case of $m = 3$ can be proved similarly. The result can be obtained through the following steps.

(1) Jobs in layer I have to be scheduled on m machines evenly. Otherwise, clearly we have $\frac{C^A}{C^*} \ge 2$.

(2) Jobs in layer II have to be scheduled on m machines evenly. Otherwise, we have $C^A = 1 + 2x_2$ and $C^* \le 1 + x_2$ from Table 3(b). Therefore $\frac{C^A}{C^*} \ge \frac{1+2x_2}{1+x_2}$.

(3) Jobs in layer III have to be scheduled on k machines in $\mathcal{M}_1$ evenly. Otherwise, we have $C^A = 1+x_2+2\times\frac{3x_2^2-1}{2+2x_2} = \frac{4x_2^2+2x_2}{1+x_2}$, while $C^* \le 2x_2$ from Table 3(b). Thus $\frac{C^A}{C^*} \ge \frac{1+2x_2}{1+x_2}$.

(4) Job in the last layer has to be scheduled on one machine in $\mathcal{M}_1$. Therefore, $C^A = 1 + x_2 + \frac{3x_2^2-1}{2+2x_2} + \frac{3x_2^2-1}{1+x_2} = \frac{11x_2^2+4x_2-1}{2+2x_2}$. However, we get $C^* \le \frac{3x_2^2-1}{1+x_2}$ from Table 3(b). Hence we have $\frac{C^A}{C^*} \ge \frac{11x_2^2+4x_2-1}{6x_2^2-2} = \frac{1+2x_2}{1+x_2}$, where the last equality is due to the definition of x_2. □

Theorem 4.3. *For $k = 1$, any algorithm has a competitive ratio of at least*

$$\begin{cases} \dfrac{1+2x_3}{1+x_3} \approx 1.824, & m = 3; \\ \dfrac{1+2x_4}{1+x_4} \approx 1.848, & m = 4, 5; \\ \dfrac{1+2x_5}{1+x_5} \approx 1.829, & m \ge 6. \end{cases}$$

where $x_3 \approx 4.686$, $x_4 \approx 5.570$ and $x_5 \approx 4.865$ are the roots of the equations $x^3 - 4x^2 - 3x - 1 = 0$, $x^3 - 5x^2 - 3x - 1 = 0$ and $x^3 - 4x^2 - 4x - 1 = 0$, respectively.

Table 3(b)
Feasible solutions for partial layers.

Partial layers	A feasible schedule for $k = 2, m = 3$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I, II	2×{I, II}	{I, II}	$1 + x_1$
I, II, III	{I, III}, {2I, III}	{3II}	$3x_1$
I, II, III, IV	{2III}, {IV}	{3I, 3II}	$\frac{5x_1^2+x_1-1}{1+x_1}$
Partial layers	A feasible schedule for $k = 2, m \geq 4$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I, II	2×{I, II}	$(m-2)$×{I, II}	$1 + x_2$
I, II, III	2×{2I, III}	2×{2II}, $(m-4)$×{I, II}	$2x_2$
I, II, III, IV	{IV}, {2III}	2×{2I, 2II}, $(m-4)$×{I, II}	$\frac{3x_2^2-1}{1+x_2}$

Table 4(a)
Illustration of layers used in Theorem 4.3.

Layer	Job hierarchy	Number of jobs	Job size		
			$m = 3$	$m = 4, 5$	$m \geq 6$
I	2	$m-1$	x_3	x_4	x_5
II	2	1	$1 + x_3$	$1 + x_4$	$1 + x_5$
III	2	$m-1$	$x_3^2 - x_3$	$x_4^2 - x_4$	x_5^2
IV	2	1	$x_3^2 + x_3$	$x_4^2 + x_4$	$x_5^2 + 2x_5$
V	1	1	$2x_3^2 - x_3$	$2x_4^2 - 2x_4$	$2x_5^2$

Table 4(b)
Feasible solutions for partial layers.

Partial layers	A feasible schedule for $m = 3$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I, II	{II}	2×{I}	$1 + x_3$
I, II, III	{2I, II}	2×{III}	$x_3^2 - x_3$
I, II, III, IV	{IV}	{2I, III}, {II, III}	$x_3^2 + x_3$
I, II, III, IV, V	{V}	{I, 2III}, {I, II, IV}	$2x_3^2 - x_3$
Partial layers	A feasible schedule for $m = 4, 5$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I, II	{II}	$(m-1)$×{I}	$1 + x_4$
I, II, III	{III}	{$(m-1)$I, II}, $(m-2)$×{III}	$\leq 1 + 5x_4$
I, II, III, IV	{IV}	{2I, III}, {II, III}, $(m-3)$×{I, III}	$x_4^2 + x_4$
I, II, III, IV, V	{V}	{2III}, {IV}, $(m-4)$×{I, III}, {3I, II, III}	$2x_4^2 - 2x_4$
Partial layers	A feasible schedule for $m \geq 6$		Makespan
	$\mathcal{M}_1$	$\mathcal{M}_2$	
I, II	{II}	$(m-1)$×{I}	$1 + x_5$
I, II, III	{II}	$(m-1)$×{I, III}	$x_5^2 + x_5$
I, II, III, IV	{IV}	{2I, III}, {II, III}, $(m-3)$×{I, III}	$x_5^2 + 2x_5$
I, II, III, IV, V	{V}	{II, IV}, {2III}, {III, 3I}, $(m-4)$×{I, III}	$2x_5^2$

Proof. The sequences used in this theorem are given in Table 4(a). We only prove the case of $m \geq 6$ thoroughly, other cases can be proved similarly. The result can be obtained through the following steps.

(1) $m-1$ jobs in layer I have to be scheduled on $m-1$ different machines of $\mathcal{M}_2$. Otherwise, one job comes with hierarchy 1 and size x_5. Thus, M_1 has two jobs scheduled on it, and $C^A \geq 2x_5$. Clearly, $C^* \leq x_5$. Thus $\frac{C^A}{C^*} \geq 2$.

(2) Job in layer II have to be scheduled on M_1. Otherwise, we have $C^A = 1 + 2x_5$ and $C^* \leq 1 + x_5$ from Table 4(b). Therefore, $\frac{C^A}{C^*} \geq \frac{2x_5+1}{x_5+1}$.

(3) $m-1$ jobs in layer III are scheduled evenly on $m-1$ machines in $\mathcal{M}_2$. In fact, if any two jobs of layer III are scheduled on the same machine, we have $C^A \geq x_5 + 2x_5^2$. On the other hand, from Table 4(b), we have $C^* \leq x_5^2 + x_5$ and $\frac{C^A}{C^*} \geq \frac{2x_5+1}{x_5+1}$. If a job of layer III is scheduled on M_1, then a job with hierarchy 1 and size $x_5^2 + x_5$ arrives. It follows that $C^A = (1 + x_5) + x_5^2 + (x_5^2 + 2x_5) = 2x_5^2 + 3x_5 + 1$ while $C^* \leq x_5^2 + 2x_5$, we have $\frac{C^A}{C^*} \geq \frac{2x_5^2+3x_5}{x_5^2+2x_5} > \frac{1+2x_5}{1+x_5}$.

(4) Job in layer IV must be scheduled on M_1. Otherwise, $\frac{C^A}{C^*} \geq \frac{2x_5^2+3x_5}{x_5^2+2x_5} > \frac{1+2x_5}{1+x_5}$.

(5) Job in the last layer have to be scheduled on M_1. Therefore, $C^A = (1 + x_5) + (x_5^2 + 2x_5) + 2x_5^2 = 3x_5^2 + 3x_5 + 1$. Meanwhile, we know $C^* \leq 2x_5^2$ from Table 4(b). Thus $\frac{C^A}{C^*} \geq \frac{3x_5^2+3x_5+1}{2x_5^2} = \frac{1+2x_5}{1+x_5}$, where the last equality holds because of the definition of x_5. □

5. Conclusion

This paper studied the online scheduling problem on m parallel identical machines with two hierarchies. There are k machines with hierarchy 1 which can process all the jobs, while the remaining $m - k$ machines with hierarchy 2 can only process jobs with hierarchy 2. We designed an online algorithm *TLS* with a competitive ratio of $1 + \frac{m^2-m}{m^2-km+k^2}$. A relatively simple algorithm *SLS* whose performance is better for some pairs of k and m was also proposed. We further studied lower bounds of the problem for different pairs of k and m.

It is left as an open problem to design an optimal algorithm for any k and m. However, it could be extremely hard work. Hence, we may divide the task into several subproblems. For example, to find lower bounds for $k \geq 3$ and $k + 1 \leq m < \frac{3}{2}(k + 1)$, and to design an optimal algorithm for cases when k is small which is quite natural in real applications. We conjecture that the competitive ratio of the optimal algorithm is no greater than 2 for any k and m.

Acknowledgements

The authors would like to thank the anonymous referee for constructive comments on an earlier draft of this paper. The second author was supported by the National Natural Science Foundation of China (90818013). The third author was supported by the National Natural Science Foundation of China (10671177, 60021201) and Zhejiang Provincial Natural Science Foundation of China (Y607079).

References

[1] S. Albers, On randomized online scheduling, in: Proceedings of the 34th ACM Symposium on Theory of Computing, 2002, pp. 134–143.
[2] Y. Azar, A. Naor, R. Rom, The competitiveness of on-line assignments, Journal of Algorithms 18 (1995) 221–237.
[3] A. Bar-Noy, A. Freund, J. Naor, On-line load balancing in a hierarchical server topology, SIAM Journal on Computing 31 (2001) 527–549.
[4] O. Chassid, L. Epstein, The hierarchical model for load balancing on two machines, Journal of Combinatorial Optimization 15 (2008) 305–314.
[5] P. Crescenzi, G. Gambosi, P. Penna, On-line algorithms for the channel assignment problem in cellular networks, Discrete Applied Mathematics 137 (2004) 237–266.
[6] B. Chen, A. van Vliet, G.J. Woeginger, New lower and upper bounds for on-line scheduling, Operations Research Letters 16 (1994) 221–230.
[7] U. Faigle, W. Kern, G. Turan, On the performance of on-line algorithm for partition problems, Acta Cybernetica 9 (1989) 107–119.
[8] R. Fleischer, M. Wahl, On-line scheduling revisited, Journal of Scheduling 3 (2000) 343–353.
[9] C. Glass, H. Kellerer, Parallel machine scheduling with job assignment restrictions, Naval Research Logistics 54 (2007) 250–257.
[10] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell System Technical Journal 45 (1966) 1563–1581.
[11] H. Hwang, S. Chang, K. Lee, Parallel machine scheduling under a grade of service provition, Computers & Operations Research 31 (2004) 2055–2061.
[12] Y.W. Jiang, Online scheduling on parallel machines with two GoS levels, Journal of Combinatorial Optimization 16 (2008) 28–38.
[13] Y.W. Jiang, Y. He, C.M. Tang, Optimal online algorithms for scheduling on two identical machines under a grade of service, Journal of Zhejiang University Science 7A (2006) 309–314.
[14] J. Park, S.Y. Chang, K. Lee, Online and semi-online scheduling of two machines under a grade of service provision, Operations Research Letters 34 (2006) 692–696.
[15] F. Rudin III, R. Chandrasekaran, Improved bounds for the online scheduling problem, SIAM Journal on Computing 32 (2003) 717–735.